Partie A : À l'écrit (40 minutes)

Exercice 1 :

Écrire le nombre 101, écrit en binaire, en nombre décimal.

Exercice 2 :

Écrire le nombre 63 en binaire.

Exercice 3 :

On se donne trois booléens a, b, c. Montrer à l'aide d'une table de vérité l'égalité suivante :

$$(a \text{ et } b) \text{ ou } (a \text{ et } c) = a \text{ et } (b \text{ ou } c)\,.$$

Exercice 4 :

Pouvez-vous expliquer en quelques phrases ce qu'est le codage ASCII, à quoi il sert, quelle est la limite de ce codage.

On finira par dire pourquoi le codage **utf8** repousse ces limites en rappelant le principe de ce codage.

Exercice 5 :

On se donne l'algorithme suivant.

```
for i in xrange(1,10):
    a = ""
    for j in xrange(1,i):
        a = a+"*"
    print a
```

Donner l'affichage de celui-ci.

Partie B : À l'ordinateur (1h15)

Le pendu est un jeu consistant à trouver un mot en devinant quelles sont les lettres qui le composent.

Déroulement Le joueur dans cet exemple s'appelle Stéphane :

L'ordinateur choisit un mot. Stéphane annonce une lettre. La lettre fait-elle partie du mot ?

— Oui : L'ordinateur affiche le mot en mettant des étoiles sur les lettres non trouvées.

— Non : L'ordinateur annonce un échec et avance le dessin du pendu.

Le jeu se poursuit jusqu'à ce que : Stéphane gagne la partie en trouvant toutes les lettres du mot ou en le devinant correctement. L'ordinateur gagne la partie après 7 échecs.

Normalement vous avez accès à trois fichiers pendue_note.py et liste.de.mots.francais.txt et image.py

Exercice 1 :

Ouvrir dans spyder le fichier pendue_note.py

Exercice 2 :

Comprendre la fonction **choisir_mot()** : Afficher un mot au hasard et donner le nombre de lettres de ce mot.

Exercice 3 :

Écrire la fonction **mot_devoile** qui prend en arguments une liste de lettres et un mot et affiche le mot composé uniquement des lettres de lettres_trouvees. Exemple si lettres_trouvees = ["a","r"] et mot="armoire", la fonction affiche **ar***r*.

Indication :

a. Initialiser une chaîne de caractères vide.

b. Pour chaque lettre du mot, vérifier que la lettre est dans la liste des lettres trouvées.

c. Si la lettre est dans la liste, ajouter la lettre à la fin du mot.

d. Sinon, ajouter le caractère * à la fin du mot.

Exercice 4 :

Modifier cette fonction pour qu'elle renvoie vrai si toutes les lettres du mot sont dans **lettres_trouvees** et faux sinon.

Indication : On part du principe que toutes les lettres sont dans le mot (donc le booléen sera vrai par défaut). À chaque fois que la lettre n'y est pas, ce booléen devient faux. *On pensera à définir une variable représentant ce booléen.*

Exercice 5 :

Écrire la fonction **lettre_presente** qui prend une lettre, un mot, une liste de lettres et un nombre comme arguments.

Si la lettre est dans le mot, on rajoute ce mot à la liste de lettres trouvées sinon on rajoute un échec.

Cette fonction renvoie la nouvelle listes de lettres trouvées, le nombre d'échecs et un booléen qui renvoie vrai si toutes les lettres du mot sont dans **lettres_trouvees** et faux sinon.

*On pensera à utiliser la fonction **mot_devoile** dans cette fonction.*

Exercice 6 :

Utiliser l'ensemble de ces fonctions pour écrire le programme du jeu du pendu.

Exercice 7 :

Normalement une potence et un pendu doivent être dessinés au fur et à mesure des échecs. En utilisant le fichier **dessin.py** et les fonctions qu'il contient, ajouter cette fonctionnalité au programme.